Module Interface Specification for Solar Water Heating Systems Incorporating Phase Change Material

Brooks MacLachlan and Spencer Smith

October 25, 2017

Contents

1 Introduction

The following document details the Module Interface Specifications for the implemented modules in a program simulating a Solar Water Heating System with Phase Change Material. It is intended to ease navigation through the program for design and maintenance purposes.

Complementary documents include the System Requirement Specifications and Module Guide.

2 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by SWHS.

Data Type	**Notation**	**Description**
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

The specification of SWHS uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, SWHS uses functions, which are defined by the data types of their inputs and outputs. Functions are described by showing their input data types separated by multiplication symbols on the left side of an arrow, and their output data type on the right side.

3 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Format Module Input Parameters Module Input Verification Module Output Format Module Output Verification Module Temperature ODEs Module Energy Equations Module Control Module
Software Decision Module	Sequence Data Structure Module ODE Solver Module Plotting Module

Table 1: Module Hierarchy

4 MIS of Control Module

4.1 Module

main

4.2 Uses

parameters (Section 5), load_params (Section 6), verify_params (Section 7), temperature (Section 8), ODE Solvers Module (Section 9), energy (Section 10), verify_output (Section 11), plot (Section 12), output (Section 13)

4.3 Syntax

4.3.1 Exported Access Programs

Name	In	Out	Exceptions
main	string	-	-

4.4 Semantics

4.4.1 State Variables

$time$: array of $\mathbb{R}$
$tempW$: array of $\mathbb{R}$
$tempP$: array of $\mathbb{R}$
$latHeat$: array of $\mathbb{R}$
eW: array of $\mathbb{R}$
eP: array of $\mathbb{R}$
$eTot$: array of $\mathbb{R}$

4.4.2 Environment Variables

win: 2D array of pixels displayed on the screen

4.4.3 Access Routine Semantics

main(s): transition: $time$, $tempW$, $tempP$, $latHeat$, eW, eP, $eTot$, win := $results[0]$, $results[1]$, $results[2]$, $results[3]$, $eW1 \| eW2 \| eW3$, $eP1 \| eP2 \| eP3$, $(\forall i \in [0..|post(eW)|-1])\ (post(eW[i]) + post(eP[i]))$, Prints information about the melting of PCM.

exception: none

5 MIS of Input Parameters Module

The secrets of this module are the data structure for input parameters, how the values are input and how the values are verified. The load and verify secrets are isolated to their own access programs.

5.1 Module

Param

5.2 Uses

N/A

5.3 Syntax

Name	**In**	**Out**	**Exceptions**
load_params	string	-	FileError
verify_params	-	$\mathbb{B}$	badLength, badDiam, badPCMVolume, badPCMAndTankVol, badPCMArea, badPCMDensity, badMeltTemp, badCoilAndInitTemp, badCoilTemp, badPCMHeatCapSolid, badPCMHeatCapLiquid, badHeatFusion, badCoilArea, badWaterDensity, badWaterHeatCap, badCoilCoeff, badPCMCoeff, badInitTemp, badFinalTime, badInitAndMeltTemp
L	-	$\mathbb{R}$	
D	-	$\mathbb{R}$	
V_P	-	$\mathbb{R}$	
A_P	-	$\mathbb{R}$	
...	-	...	

5.4 Semantics

5.4.1 Environment Variables

f: sequence of string *#f[i] is the ith string in the text file f*

5.4.2 State Variables

# From R1
L: $\mathbb{R}$
D: $\mathbb{R}$
V_P: $\mathbb{R}$
A_P: $\mathbb{R}$
ρ_P : $\mathbb{R}$
T_{melt}^P: $\mathbb{R}$
C_P^S: $\mathbb{R}$
C_P^L: $\mathbb{R}$
H_f: $\mathbb{R}$
A_C: $\mathbb{R}$
T_C: $\mathbb{R}$
ρ_W: $\mathbb{R}$
C_W: $\mathbb{R}$
h_C: $\mathbb{R}$
h_P: $\mathbb{R}$
T_{init}: $\mathbb{R}$
t_{step}: $\mathbb{R}$
t_{final}: $\mathbb{R}$
$AbsTol$: $\mathbb{R}$
$RelTol$: $\mathbb{R}$
$ConsTol$: $\mathbb{R}$

# From R2
V_{tank}: $\mathbb{R}$
m_W: $\mathbb{R}$
m_P: $\mathbb{R}$

# From R3
τ_W: $\mathbb{R}$

η: $\mathbb{R}$
τ_P^S: $\mathbb{R}$
τ_P^L: $\mathbb{R}$

# To Support IM4
$E_{P\text{melt}}^{\text{init}}$: $\mathbb{R}$
$E_{P\text{melt}}^{\text{all}}$: $\mathbb{R}$

# To Support Testing
τ_P^L m_W^{noPCM}: $\mathbb{R}$
τ_W^{noPCM}: $\mathbb{R}$

5.4.3 Assumptions

load_params will be called before the values of any state variables will be accessed.

5.4.4 Access Routine Semantics

init():

- transition: L, diam, Vp, Ap, ..., tau_w_no_PCM := 0, 0, 0, 0, ..., 0
- output: *out* := self
- exception: none

getL():

- output: *out* := L
- exception: none

get_diam():

- output: *out* := diam
- exception: none

getVp():

- output: *out* := Vp
- exception: none

getAp():

- output: *out* := Ap
- exception: none

...

get_tau_w_no_PCM():

- output: *out* := tau_w_no_PCM
- exception: none

setL(x):

- transition: L := x
- exception: none

set_diam(x):

- transition: diam := x
- exception: none

setVp(x):

- transition: Vp := x
- exception: none

setAp(x):

- transition: Ap := x
- exception: none

...

set_tau_w_no_PCM(x):

- transition: tau_w_no_PCM := x
- exception: none

5.5 Considerations

The value of each state variable can be accessed through its name (getter). An access program is available for each state variable. There are no setters for the state variables, since the values will be set and checked by load params and not changed for the life of the program.

6 MIS of Input Format Module

6.1 Module

Load_params

6.2 Uses

Param (Section 5)

6.3 Syntax

6.4 Exported Access Programs

Name	In	Out	Exceptions
load_params	string	-	-

6.5 Semantics

6.5.1 Environment Variables

f: sequence of string #*f[i] is the ith string in the text file f*

6.5.2 Assumptions

The input string corresponds to an existing filename. The name will be relative to the current directory. The input file is assumed to be formatted correctly. The file contains the string equivalents of the numeric values for each input parameter in order, each on a new line. The order is the same as in the table in R1 of the SRS. Any comments in the input file should be denoted with a '#' symbol.

6.5.3 Access Routine Semantics

load_params(s):

- transition: The filename s is first associated with the file f. File f is then used to modify the state of Param (Section 5) as follows:

1. Param.init()
2. Read data sequentially from f to populate the state variables of Param from L to ConsTol.
3. Calculate the derived quantities in Param as follows:
   - Param.setVt(calcVt(Param.getL(), Param.get_diam()))
   - Param.setMw(calcMw(Param.getVp(), Param.get_rho_w(), Param.getVt()))
   - Param.set_tau_w(calcTauw(Param.getMw(), Param.getC_w(), Param.get_hc(), Param.getAc()))
   - Param.set_eta(calcEta(Param.get_hp(), Param.getAp(), Param.get_hc(), Param.getAc()))
   - Param.setMp(calcMp(Param.get_rho_p(), Param.getVp()))
   - Param.set_tau_ps(calcTaups(Param.getMp(), Param.getC_ps(), Param.get_hp(), Param.getAp()))
   - Param.set_taul_pl(calcTaupl(Param.getMp(), Param.getC_pl(), Param.get_hp(), Param.getAp()))
   - Param.setEpmelt_init(calcEpmeltinit(Param.getC_ps(), Param.getMp(), Param.getTmelt(), Param.getTinit()))
   - Param.setEp_melt3(calcEpmelt3(Param.getHf(), Param.getMp()))
   - Param.setMw_noPCM(calcMwno(Param.get_rho_w, Param.getVt()))
   - Param.set_tau_no_PCM(calcTauwnoPCM(Param.getMw_noPCM(), Param.getC_w(), Param.get_hc(), Param.getAc()))

- exception: none

6.5.4 Local Functions

calcVt: $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
calcVt(L, d) $\equiv \pi \times L \times (\frac{d}{2})^2$

calcMw: $\mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
calcMw(V_p, ρ_w, V_t) $\equiv \rho_w(V_t - V_p)$

calcTauw: $\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
calcTauw(m_w, C_w, h_c, A_c) $\equiv \frac{m_w C_w}{A_c h_c}$

calcEta: $\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$

calcEta$(h_p,\ A_p,\ h_c,\ A_c) \equiv \frac{h_p A_p}{h_c A_c}$

calcMp: $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
calcMp$(\rho_p,\ V_p) \equiv \rho_p V_p$

calcTaups: $\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
calcTaups$(M_p,\ C_{ps},\ h_p,\ A_p) \equiv \frac{M_p C_{ps}}{h_p A_p}$

calcTaupl: $\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
calcTaupl$(M_p,\ C_{pl},\ h_p,\ A_p) \equiv \frac{M_p C_{pl}}{h_p A_p}$

calcEpmeltinit: $\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
calcEpmeltinit$(C_{ps},\ M_p,\ T_{\text{melt}},\ T_{\text{init}}) \equiv C_{ps} M_p (T_{\text{melt}} - T_{\text{init}})$

calcEpmelt3: $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
calcEpmelt3$(H_f,\ M_p) \equiv H_f M_p$

calcMwnoPCM: $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
calcMwnoPCM$(\rho_w,\ V_t) \equiv \rho_w V_t$

calcTauwnoPCM: $\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
calcTauwnoPCM$(M_{w\text{noPCM}},\ C_w,\ h_c,\ A_c) \equiv \frac{M_{w\text{noPCM}} C_w}{h_c A_c}$

7 MIS of Input Verification Module

7.1 Module

verify_params

7.2 Uses

Param (Section 5)

7.3 Syntax

7.3.1 Exported Access Programs

Name	In	Out	Exceptions
verify_valid	-	-	badLength, badDiam, badPCMVolume, badPCMAndTankVol, badPCMArea, badPCMDensity, badMeltTemp, badCoilAndInitTemp, badCoilTemp, badPCMHeatCapSolid, badPCMHeatCapLiquid, badHeatFusion, badCoilArea, badWaterDensity, badWaterHeatCap, badCoilCoeff, badPCMCoeff, badInitTemp, badFinalTime, badInitAndMeltTemp
verify_recommend	-	-	-

7.4 Semantics

7.4.1 Assumptions

All of the fields Param have been assigned values before any of the access routines for this module are called.

7.4.2 Access Routine Semantics

verify_valid():

- transition: none

- exceptions: exc := (
  Param.getL() $\leq 0 \Rightarrow$ badLength |
  Param.get_diam() $\leq 0 \Rightarrow$ badDiam |
  Params.get_Vp() $\leq 0 \Rightarrow$ badPCMVolume |
  Params.getVp() $\geq$ Params.Vt $\Rightarrow$ badPCMAndTankVol |
  Params.getAp() $\leq 0 \Rightarrow$ badPCMArea |
  Params.get_rho_p() $\leq 0 \Rightarrow$ badPCMDensity |
  Params.getTmelt() $\leq 0 \Rightarrow$ badMeltTemp |
  Params.getTmelt() $\geq$ Params.getTc() $\Rightarrow$ badMeltTemp |
  Params.getTc() $\leq$ Params.getTinit() $\Rightarrow$ badCoilAndInitTemp |
  Params.getTc() $\geq 100\vee$ Params.getTc() $\leq 0 \Rightarrow$ badCoilTemp |
  Params.getC_ps() $\leq 0 \Rightarrow$ badPCMHeatCapSolid |
  Params.getC_pl() $\leq 0 \Rightarrow$ badPCMHeatCapLiquid |
  Params.getHf() $\leq 0 \Rightarrow$ badHeatFusion |
  Params.getAc() $\leq 0 \Rightarrow$ badCoilArea |
  Params.get_rho()_w $\leq 0 \Rightarrow$ badWaterDensity |
  Params.getC_w() $\leq 0 \Rightarrow$ badWaterHeatCap |
  Params.get_hc() $\leq 0 \Rightarrow$ badCoilCoeff |
  Params.get_hp() $\leq 0 \Rightarrow$ badPCMCoeff |
  Params.getTinit() $\leq 0\vee$ Params.getTinit() $\geq 100 \Rightarrow$ badInitTemp |
  Params.get_tfinal() $\leq 0 \Rightarrow$ badFinalTime |
  Params.getTinit() $\geq$ Params.getTmelt() $\Rightarrow$ badInitAndMeltTemp)

verify_recommend():

- transition: none

- exceptions: exc := (
  Params.getL() $< 0.1\vee$ Params.getL() $> 50 \Rightarrow$ warnLength |
  Params.getdiam() / Params.getL() $< 0.002\vee$ Params.getdiam() / Params.getL() $> 200 \Rightarrow$ warnDiam |
  Params.getVp() $<$ Params.getVt() $\times 10^-6 \Rightarrow$ warnPCMVol |
  Params.getVp() $>$ Params.getAp() $\vee$ Params.getAp $>$ (2/0.001)$\times$ Params.getVp() $\Rightarrow$ warnVolArea |
  (Params.get_rho_p() ≤ 500) $\vee$ (Params.get_rho_p() ≥ 20000) $\Rightarrow$ warn-PCMDensity | ...)
  # Need to continue for the rest of the example - tabular form? # Should add a module (Configuration Module) to store symbolic constants

7.5 Considerations

See Appendix (Section 14) for the complete list of exceptions and associated error messages.

8 MIS of Temperature ODEs Module

8.1 Module

temperature

8.2 Uses

Param (Section 5)

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
ODE_SolidPCM	–	sequence[2] of $\mathbb{R} \rightarrow$ sequence[2] of $\mathbb{R}$	-
ODE_MeltingPCM	–	sequence[3] of $\mathbb{R} \rightarrow$ sequence[3] of $\mathbb{R}$	-
ODE_LiquidPCM	–	sequence[2] of $\mathbb{R} \rightarrow$ sequence[2] of $\mathbb{R}$	-
event_StartMelt	–	sequence[2] of $\mathbb{R} \rightarrow \mathbb{R}$	-
event_EndMelt	–	sequence[3] of $\mathbb{R} \rightarrow \mathbb{R}$	-

8.4 Semantics

8.4.1 State Variables

none

8.4.2 Assumptions

none

8.4.3 Access Routine Semantics

ODE_SolidPCM():

- output: $out := \frac{d}{dt}\begin{bmatrix} T_W \\ T_P \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))] \\ \frac{1}{\tau_P^S}(T_W(t) - T_P(t)) \end{bmatrix}$

- exception: none

ODE_MeltingPCM():

- output: $out := \dfrac{d}{dt}\begin{bmatrix} T_W \\ T_P \\ Q_P \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))] \\ 0 \\ h_P A_P (T_W(t) - T_{\text{melt}}^P) \end{bmatrix}$
- exception: none

ODE_LiquidPCM():

- output: $out := \dfrac{d}{dt}\begin{bmatrix} T_W \\ T_P \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))] \\ \frac{1}{\tau_P^L}(T_W(t) - T_P(t)) \end{bmatrix}$
- exception: none

event_StartMelt():

- output: $out := g([T_W, T_P]^T) = T_{\text{melt}}^P - T_P$
- exception: none

event_EndMelt():

- output: $out := g([T_W, T_P, Q_P]^T) = 1 - \phi$, where $\phi = \frac{Q_P}{H_f m_P}$
- exception: none

9 MIS of ODE Solver Module

9.1 Module

ODE Solver Module

9.2 Uses

N/A

9.3 Syntax

9.3.1 Exported Constants

$MaxStep$: natural number
N: natural number

9.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
solve	function, array of $\mathbb{R}$, array of $\mathbb{R}$, function, $\mathbb{R}$, $\mathbb{R}$	array of $\mathbb{R}$ (N of them)	ODE_BAD_INPUT, ODE_MAXSTEP, ODE_ACCURACY

9.4 Semantics

9.4.1 State Variables

$results$: array of $\mathbb{R}$ (N of them)

9.4.2 Access Routine Semantics

solve(f, *domain*, *ics*, *events*, *abstol*, *reltol*)	output:	*out* := *results*, where *results* holds the solution to the ODE system generated by the solver.
	exceptions:	*exc* := (Invalid input parameters ⇒ ODE_BAD_INPUT | *MaxStep* steps taken and no solution found ⇒ ODE_MAXSTEP | *reltol* and *abstol* not satisfied for a step ⇒ ODE_ACCURACY)

10 MIS of Energy Module

10.1 Module

energy

10.2 Uses

Param (Section 5)

10.3 Syntax

10.3.1 External Access Programs

Name	In	Out	Exceptions
energy1Wat	array of $\mathbb{R}$, parameters	array of $\mathbb{R}$	-
energy1PCM	array of $\mathbb{R}$, parameters	array of $\mathbb{R}$	-
energy2Wat	array of $\mathbb{R}$, parameters	array of $\mathbb{R}$	-
energy2PCM	array of $\mathbb{R}$, parameters	array of $\mathbb{R}$	-
energy3Wat	array of $\mathbb{R}$, parameters	array of $\mathbb{R}$	-
energy3PCM	array of $\mathbb{R}$, parameters	array of $\mathbb{R}$	-

10.4 Semantics

10.4.1 State Variables

$eW1$: array of $\mathbb{R}$
$eP1$: array of $\mathbb{R}$
$eW2$: array of $\mathbb{R}$
$eP2$: array of $\mathbb{R}$
$eW3$: array of $\mathbb{R}$
$eP3$: array of $\mathbb{R}$

10.4.2 Assumptions

All of the fields of the input parameters structure have been assigned a value. The values have been properly constrained.

10.4.3 Access Routine Semantics

energy1Wat($Tw1$, $params$):

transition: $(\forall i \in [0..|Tw1|-1])\ (eW1[i] :=$ watEnergy$(Tw1[i], params))$

output: $out := eW1$

exception: none

energy1PCM($Tp1$, $params$):

transition: $(\forall i \in [0..|Tp1|-1])\ (eP1[i] :=$ pcmEnergy1$(Tp1[i], params))$

output: $out := eP1$

exception: none

energy2Wat($Tw2$, $params$):

transition: $(\forall i \in [0..|Tw2|-1])\ (eW2[i] :=$ watEnergy$(Tw2[i], params))$

output: $out := eW2$

exception: none

energy2PCM($Qp2$, $params$):

transition: $(\forall i \in [0..|Qp2|-1])\ (eP2[i] :=$ pcmEnergy2$(Qp2[i], params))$

output: $out := eP2$

exception: none

energy3Wat($Tw3$, $params$):

transition: $(\forall i \in [0..|Tw3|-1])\ (eW3[i] :=$ watEnergy$(Tw3[i], params))$

output: $out := eW3$

exception: none

energy3PCM($Tp3$, $params$):

transition: $(\forall i \in [0..|Tp3|-1])\ (eP3[i] :=$ pcmEnergy3$(Tp3[i], params))$

output: $out := eP3$

exception: none

10.4.4 Local Functions

watEnergy: $\mathbb{R} \times$ parameters $\rightarrow \mathbb{R}$
watEnergy(Tw, $params$) $\equiv params.C\_w \times params.Mw \times (Tw - params.Tinit)$

pcmEnergy1: $\mathbb{R} \times$ parameters $\rightarrow \mathbb{R}$
pcmEnergy1(Tp, $params$) $\equiv params.C\_ps \times params.Mp \times (Tp - params.Tinit)$

pcmEnergy2: $\mathbb{R} \times$ parameters $\rightarrow \mathbb{R}$
pcmEnergy2(Qp, $params$) $\equiv params.Epmelt\_init + Qp$

pcmEnergy3: $\mathbb{R} \times$ parameters $\rightarrow \mathbb{R}$
pcmEnergy3(Tp, $params$) $\equiv params.Epmelt\_init + params.Ep\_melt3 + params.C\_pl \times params.Mp \times (Tp - params.Tmelt)$

11 MIS of Output Verification Module

11.1 Module

verify_output

11.2 Uses

Param (Section 5)

11.3 Syntax

11.3.1 Exported Access Programs

Name	In	Out	Exceptions
verify_output	array of $\mathbb{R}$, array of $\mathbb{R}$, array of $\mathbb{R}$, array of $\mathbb{R}$, array of $\mathbb{R}$, parameters	-	-

11.4 Semantics

11.4.1 State Variables

$expEPCM$: array of $\mathbb{R}$
$expEWat$: array of $\mathbb{R}$
$errorWater$: $\mathbb{R}$
$errorPCM$: $\mathbb{R}$

11.4.2 Environment Variables

win: 2D array of pixels displayed on the screen

11.4.3 Local Variables

11.4.4 Assumptions

All of the fields of the input parameters structure have been assigned a value. The values have been properly constrained. The input arrays are not empty.

11.4.5 Access Routine Semantics

verify_output(t, Tw, Tp, Ew, Ep, $params$):

transition: $expEPCM$, $expEWat$, $errorWater$, $errorPCM$, win := ($\forall i \in [1..|t| - 1]$) (expectedEp(traprule(delta($t[i-1]$, $t[i]$), $Tw[i]$, $Tp[i]$, $Tw[i-1]$, $Tp[i-1]$), $params$)), ($\forall i \in [1..|t| - 1]$) (expectedEw (expectedEc(traprule(delta($t[i-1]$, $t[i]$), $params.Tc$, $Tw[i]$, $params.Tc$, $Tw[i-1]$), $params$), post($expEPCM$))), error(sum(post($expEWat$)), $Ew[|Ew| - 1]$), error(sum(post($expEPCM$)), $Ep[|Ep| - 1]$), ($errorWater > ConsTol \vee errorPCM > ConsTol \Rightarrow$ Prints warning message(s))

exception: ($errorWater > ConsTol \Rightarrow$ warnWaterError | $errorPCM > ConsTol \Rightarrow$ warnPCMError) These exceptions do not terminate the program.

11.4.6 Local Functions

delta: $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
delta($t1$, $t2$) $\equiv t2 - t1$

traprule: $\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
traprule(t, $A1$, $B1$, $A2$, $B2$) $\equiv t \times (A1 - B1 + A2 - B2)/2$

expectedEc: $\mathbb{R}$ × parameters $\rightarrow \mathbb{R}$
expectedEc(c, $params$) $\equiv params.hc \times params.Ac \times c$

expectedEp: $\mathbb{R} \times$ parameters $\rightarrow \mathbb{R}$
expectedEp(p, $params$) $\equiv params.hp \times params.Ap \times p$

expectedEw: $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
expectedEw(Ec, Ep) $\equiv Ec - Ep$

sum: array of $\mathbb{R}$s $\rightarrow \mathbb{R}$
sum(a) $\equiv \sum_{i=0}^{|a|-1} a[i]$

error: $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
error(exp, act) $\equiv \frac{|exp-act|}{act} \times 100$

12 MIS of Plotting Module

12.1 Module

plot

12.2 Uses

N/A

12.3 Syntax

12.3.1 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
plot	array of $\mathbb{R}$, array of $\mathbb{R}$, array of $\mathbb{R}$, array of $\mathbb{R}$, array of $\mathbb{R}$, string	-	-

12.4 Semantics

12.4.1 State Variables

$plotFilename$: string

12.4.2 Environment Variables

directory: The current directory of files from which the program is run.

12.4.3 Assumptions

The input arrays are all of the same size.

12.4.4 Access Routine Semantics

plot(t, Tw, Tp, Ew, Ep, $filename$):	transition:	*directory*: writes a .png file named *plotFilename* containing the graphs of the simulation results.
	exception:	none

13 MIS of Output Module

13.1 Module

output

13.2 Uses

Param (Section 5)

13.3 Syntax

13.3.1 Exported Constants

max_width: integer

13.3.2 Exported Access Program

Name	**In**	**Out**	**Exceptions**
output	string, array of $\mathbb{R}$, array of $\mathbb{R}$, array of $\mathbb{R}$, array of $\mathbb{R}$, array of $\mathbb{R}$, array of $\mathbb{R}$, parameters	-	-

13.4 Semantics

13.4.1 State Variables

outFilename: string

13.4.2 Environment Variables

directory: The current directory of files from which the program is run.

13.4.3 Access Routine Semantics

output($params$, t, Tw, Tp, Ew, Ep, $ETot$, $filename$):

- transition: $directory$: writes a .txt file named $outFilename$ containing the input parameters, calculated parameters, and results of the simulation.
- exception: none

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995.

14 Appendix

Table 2: Possible Exceptions

Message ID	**Error Message**
badLength	Error: Tank length must be > 0
badDiam	Error: Tank diameter must be > 0
badPCMVolume	Error: PCM volume must be > 0
badPCMAndTankVol	Error: PCM volume must be $<$ tank volume
badPCMArea	Error: PCM area must be > 0
badPCMDensity	Error: rho_p must be > 0
badMeltTemp	Error: Tmelt must be > 0 and $< Tc$
badCoilAndInitTemp	Error: Tc must be $>$ Tinit
badCoilTemp	Error: Tc must be > 0 and < 100
badPCMHeatCapSolid	Error: C_ps must be > 0
badPCMHeatCapLiquid	Error: C_pl must be > 0
badHeatFusion	Error: Hf must be > 0
badCoilArea	Error: Ac must be > 0
badWaterDensity	Error: rho_w must be > 0
badWaterHeatCap	Error: C_w must be > 0
badCoilCoeff	Error: hc must be > 0
badPCMCoeff	Error: hp must be > 0
badInitTemp	Error: Tinit must be > 0 and < 100
badFinalTime	Error: tfinal must be > 0
badInitAndMeltTemp	Error: Tinit must be $<$ Tmelt
ODE_ACCURACY	*reltol* and *abstol* were not satisfied by the ODE solver for a given solution step.
ODE_BAD_INPUT	Invalid input to ODE solver
ODE_MAXSTEP	ODE solver took $MaxStep$ steps and did not find solution
warnLength	Warning: It is recommended that 0.1 <= L <= 50
warnDiam	Warning: It is recommended that 0.002 <= D/L <= 200

warnPCMVol	Warning: It is recommended that Vp be >= 0.0001% of Vt
warnVolArea	Warning: It is recommended that Vp <= Ap <= (2/0.001) * Vp
warnPCMDensity	Warning: It is recommended that 500 < rho_p < 20000
warnPCMHeatCapSolid	Warning: It is recommended that 100 < C_ps < 4000
warnPCMHeatCapLiquid	Warning: It is recommended that 100 < C_pl < 5000
warnCoilArea	Warning: It is recommended that Ac <= pi * (D/2) ∧ 2
warnWaterDensity	Warning: It is recommended that 950 < rho_w <= 1000
warnWaterHeatCap	Warning: It is recommended that 4170 < C_w < 4210
warnCoilCoeff	Warning: It is recommended that 10 < hc < 10000
warnPCMCoeff	Warning: It is recommended that 10 < hp < 10000
warnFinalTime	Warning: It is recommended that 0 < tfinal < 86400
warnWaterError	Warning: There is greater than x% relative error between the energy in the water output and the expected output based on the law of conservation of energy. (Where x is the value of $ConsTol$)
warnPCMError	Warning: There is greater than x% relative error between the energy in the PCM output and the expected output based on the law of conservation of energy. (Where x is the value of $ConsTol$)