Module Interface Specification for Software for Fraction Solid

Spencer Smith, Malavika Srinivasan and Vincent Cherk

November 21, 2018

1 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://gitlab.cas.mcmaster.ca/SEforSC/se4sc/tree/git-svn/GradStudents/Malavika/SRSSFS

Contents

2 Introduction

The following document details the Module Interface Specifications for the implemented modules in a program simulating the fraction of solid formed during the casting process over time. It is intended to ease navigation through the program for design and maintenance purposes.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://gitlab.cas.mcmaster.ca/SEforSC/se4sc/tree/git-svn/GradStudents/Malavika.

The specification is given in terms of functions, rather than sequences. For instance, the predicted fraction of the solid is given as a function of time ($\mathbb{R} \to \mathbb{R}$), not as a sequence ($\mathbb{R}^n$). This approach is more straightforward for the specification, but in the implementation stage, it will likely be necessary to introduce to introduce a sequence, assuming that a numerical solver is used for the system of ODEs.

3 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 SFS.

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 [0, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

The specification of SFS 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. SFS also uses some user defined data types, as given in the MIS that follows.

4 Module Decomposition

The following table is taken directly from the Module Guide document for this project. [This table has to be updated when the MG is complete. —SS]

Level 1	Level 2
Hardware-Hiding	
Behaviour-Hiding	Control Module Configuration Module Experiment Module Parameter Specification Module Temperature Module Phase Change Points Module Calculation Module Load Module Output Verification Module Output Module
Software Decision	Piecewise Data Structure Module Sequence Services Module Interpolation Module Linear Regression Module ODE Solver Module Plot Module

Table 1: Module Hierarchy

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. URL http://citeseer.ist.psu.edu/428727.html.

5 MIS of Control Module

5.1 Module

main [Control is actually handled by the GUI. See documentation by capstone team —SS]

5.2 Uses

config(6), expt(7), specParam(8), temp(9), calc(11), verifyOutput(13), Output(14), interp(16), LinerReg(17), ODE(18).

5.3 Syntax

5.3.1 Exported Access Programs

Name	In	Out	Exceptions
main	-	-	-

5.4 Semantics

5.4.1 Environment Variables

mode: $\mathbb{R}$
inputFile: sequence of string

5.4.2 State Variables

None

5.4.3 Access Routine Semantics

main():

- transition:

Mode = Config:

Step 1: Obtain the Parameter specification values by using parameter specification module (8):

Get (filenameParam: string)

Step 2: Modify the state of the variables for the config(Section 6) module by:

Get (filenameIn: string)

Step 3: Verify the state variables of the config module against their range obtained in step 1:

Mode = Calib:

Step 4: Get the material properties for the known alloy and modify the state of the variables for the experiment module (7) by:

Get (filenameExp: string) [This is for the material properties of the known alloy. —Malavika]

Step 5: Modify the state variables (coefficients, T_L, T_S) for the Temperature module by obtaining temperature data of the calibration experiment and verify them as an when read.

Get (filenamecalib: string)

Step 6: Modify the state variables of the experiment module (7) by finding h using calculate_h() from calculate module (11) and Temperature module (9).

Mode = Calc:

Step 7: Modify the state variables (coefficients, T_L, T_S) for the Temperature module by obtaining temperature data of the calculation experiment and verify them as an when read.

Get (filenamecalc: string)

Step 8: Modify the state variables of the calculate module (11) by finding α_b and α_e using the methods calculate_α_b() and calculate_α_e().

Step 9: Obtain the necessary inputs for calculating fraction solid - L. [Is L supposed to be user input? —Malavika] [I'm not sure what motivates this question. R14 lists L (IM4) as an input. Is something potentially incorrect in the SRS? —SS][No Dr.Smith, SRS

says "input L" but does not say whether it is from user or to be calculated.I guess it should be user input. I get this question because in SRS, L is listed along with α, ρ and C_v , some of which are supposed to be calculated. —Malavika]

Step 10: Calculate fs using the method calculate_FS() in calculate module (11), ODE Solver module (18) and Temperature module (9).

Step 11: Verify the fraction solid output using the output verification module (13).

Step 12: Output the fraction solid results of calibration and calculation to respective output files using the output module (14).

Step 13: Plot the fraction solid with respect to time, temperature and cooling rate. Plotting will be handled by as per requirement using plotting module (19).

6 MIS of Configuration Module

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

6.1 Module

config

6.2 Uses

SpecParam (Section 8)

6.3 Syntax

Name	In	Out	Exceptions
load_params	string	-	FileError
verify_params	-	-	ValueError
H	-	$\mathbb{R}$	
D	-	$\mathbb{R}$	
n	-	$\mathbb{N}$	
y_{TC}	-	$\mathbb{R}^n$	
dt	-	$\mathbb{R}$	
dy	-	$\mathbb{R}$	

6.4 Semantics

6.4.1 Environment Variables

inputFile: sequence of string

6.4.2 State Variables

From R1
H: $\mathbb{R}$
D: $\mathbb{R}$
n: $\mathbb{R}$
y_{TC}: $\mathbb{R}^n$
dt: $\mathbb{R}$
dy: $\mathbb{R}$

6.4.3 Assumptions

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

6.4.4 Access Routine Semantics

config.H:

- output: $out := H$
- exception: None

config.D:

- output: $out := D$
- exception: None

config.n:

- output: $out := n$
- exception: None

config.dt:

- output: $out := dt$
- exception: None

config.dy:

- output: $out := dy$
- exception: None

config.y_{TC}:

- output: $out := y_{TC}$
- exception: None

load_params(s):

- transition: inputFile is used to modify the state variables using the following procedural specification:

    1. Read data sequentially from inputFile to populate the state variables from R1 (H to y_{TC}).
    2. verify_params()

- exception: exc := a file name s cannot be found OR the format of inputFile is incorrect $\Rightarrow$ FileError

verify_params():

- exception: exc :=

Name	**Exception**
$\neg(H_{\min} \leq H \leq H_{\max})$	$\Rightarrow$ ValueError
$\neg(D_{\min} \leq D \leq D_{\max})$	$\Rightarrow$ ValueError
$n > n_{max}$	$\Rightarrow$ ValueError
$\neg(dt_{\min} \leq dt \leq dt_{\max})$	$\Rightarrow$ ValueError
$\neg(\forall(i : \mathbb{N} \mid i \in [1..n-1] \cdot 0 \leq y_{TCi} < H \wedge y_{TCi} < y_{TCi+1}) \wedge y_{TCn} \leq H)$	$\Rightarrow$ ValueError

This table is based on the information in Table 1 (Input Variables for Configuration Mode) from the SRS.

6.5 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 the load_params access program, and the values will not changed for the life of the program.

For each of the possible ValueError exceptions, a string describing the invalid input should be attached the exception.

7 MIS of Experiment Module

7.1 Template Module

expt

7.2 Uses

SpecParam (Section 8)

7.3 Syntax

Name	In	Out	Exceptions
load_params_calib	string	-	FileError
load_params_calc	string	-	FileError
Getters			
k_S	-	$\mathbb{R}$	
k_L	-	$\mathbb{R}$	
C_p^L	-	$\mathbb{R}$	
C_p^S	-	$\mathbb{R}$	
ρ_S	-	$\mathbb{R}$	
ρ_L	-	$\mathbb{R}$	
α_S	-	$\mathbb{R}$	
α_L	-	$\mathbb{R}$	
T_{env}	-	$\mathbb{R}$	
h	-	$\mathbb{R}$	
Setters			
k_S	$\mathbb{R}$		ValueError
k_L	$\mathbb{R}$		ValueError
C_p^L	$\mathbb{R}$		ValueError
C_p^S	$\mathbb{R}$		ValueError
ρ_S	$\mathbb{R}$		ValueError
ρ_L	$\mathbb{R}$		ValueError
α_S	$\mathbb{R}$		ValueError
α_L	$\mathbb{R}$		ValueError
T_{env}	$\mathbb{R}$		ValueError
h	$\mathbb{R}$		ValueError

7.4 Semantics

7.4.1 State Variables

$k_S : \mathbb{R}$
$k_L : \mathbb{R}$
$C_p^L : \mathbb{R}$
$C_p^S : \mathbb{R}$

$\rho_S : \mathbb{R}$
$\rho_L : \mathbb{R}$
$\alpha_S : \mathbb{R}$
$\alpha_L : \mathbb{R}$
$T_{\text{env}} : \mathbb{R}$
h: $\mathbb{R}$

7.4.2 Assumptions

A setter will be called for a given state variable before the corresponding getter is called.

7.4.3 Access Routine Semantics

load_params_calib(s):

- transition: inputFile1 is used to modify the state variables using the following procedural specification:

    1. Read data sequentially from inputFile to populate the state variables from R7 (k_S to T_{env}).
    2. Use setters defined in this module, so that appropriate exceptions will be generated

- exception: exc := a file name s cannot be found OR the format of inputFile is incorrect $\Rightarrow$ FileError

load_params_calc(s):

- transition: inputFile2 is used to modify the state variables using the following procedural specification:

    1. Read data sequentially from inputFile to populate the state variables from R7 (k_S to h).
    2. Use setters defined in this module, so that appropriate exceptions will be generated

- exception: exc := a file name s cannot be found OR the format of inputFile is incorrect $\Rightarrow$ FileError

Getters

expt.k_S:

- output: $out := k_S$
- exception: none

expt.k_L:

- output: $out := k_L$
- exception: none

expt.C_p^L:

- output: $out := C_p^L$
- exception: none

expt.C_p^S:

- output: $out := C_p^S$
- exception: none

expt.ρ_S:

- output: $out := \rho_S$
- exception: none

expt.ρ_L:

- output: $out := \rho_L$
- exception: none

expt.α_S:

- output: $out := \alpha_S$
- exception: none

expt.α_L:

- output: $out := \alpha_L$
- exception: none

expt.T_{env}:

- output: $out := T_{\text{env}}$
- exception: none

expt.h:

- output: $out := h$

- exception: none

Setters

There is also a corresponding setter for each state variable. When setting the values the validity will be checked, following Table 3 (Input Variables for Material Properties and Operating Conditions) in the SRS. Any invalid inputs will generate a Value Error with a corresponding string message describing the reason the data is considered invalid.

7.5 Considerations

The state variables for the material properties and processing conditions are currently constants. In the future some of the variables may be changed to have function types ($\mathbb{R} \to \mathbb{R}$).

8 MIS of Parameter Specification Module

8.1 Module

specParam

8.2 Uses

None

8.3 Syntax

8.3.1 Exported constants

#From the Table 6 in SRS
AbsZero := 273.15
$H_{\min} := 0.001$
$H_{\max} := 100$
$D_{\min} := 0.001$
$D_{\max} := 100$
$m_{\max} := 1000$
$dt_{\min} := 0.0001$
$dt_{\max} := 1000$
$n_{\max} := 1 \times 10^4$
$T_{\min} := 100$
$T_{\max} := 1 \times 10^4$
$k_{S}\min := 0.001$
$k_{S}\max := 1 \times 10^5$
$k_{L}\min := 0.001$
$k_{L}\max := 1 \times 10^5$
$\rho_{S}\min := 0.001$
$\rho_{S}\max := 1 \times 10^4$
$\rho_{L}\min := 0.001$
$\rho_{L}\max := 1 \times 10^4$
$C_{P}^{S}\min := 1 \times 10^{-4}$
$C_{P}^{S}\max := 1 \times 10^4$
$C_{P}^{L}\min := 1 \times 10^{-4}$
$C_{P}^{L}\max := 1 \times 10^4$
$C_{v}^{L}\min := 1 \times 10^{-4}$
$C_{v}^{L}\max := 1 \times 10^4$
$C_{v}^{S}\min := 1 \times 10^{-4}$
$C_{v}^{S}\max := 1 \times 10^4$
$L_{\min} := 0.001$
$L_{\max} := 1 \times 10^{10}$

$T_{\text{env}_{\text{min}}} := 0$
$T_{\text{env}_{\text{max}}} := 100$

8.3.2 Assumptions

None

8.3.3 Access Routine Semantics

N/A

9 MIS of Temperature Module

9.1 Module

TData

9.2 Uses

config, PiecewiseADT for PiecewiseT, Load, SeqServices

Exported Constants

$\Delta t = 1 \times 10^{-5}$
$\Delta y = 1 \times 10^{-3}$

9.3 Syntax

9.3.1 Exported Access Programs

Name	In	Out	Exceptions
TData_init			
TData_add	s : PiecewiseT, $y : \mathbb{R}$, $t_L^* : \mathbb{R}$, $T_L^* : \mathbb{R}$, $t_S^* : \mathbb{R}$, $T_S^* : \mathbb{R}$		IndepNotAscending
TData_rm	$i : \mathbb{N}$		InvalidIndex
TData_getC	$i : \mathbb{N}$	PiecewiseT	InvalidIndex
TData_getLiq	$i : \mathbb{N}$	$\mathbb{R}$, $\mathbb{R}$	InvalidIndex
TData_getSol	$i : \mathbb{N}$	$\mathbb{R}$, $\mathbb{R}$	InvalidIndex
TData_slice	$t : \mathbb{R}$	seq of $\mathbb{R}$, seq of $\mathbb{R}$	
TData_breakPts		seq of $\mathbb{R}$, seq of $\mathbb{R}$, seq of $\mathbb{R}$	
TData_T	$y : \mathbb{R}$, $t : \mathbb{R}$	$\mathbb{R}$	OutOfDomain
TData_dTdt	$y : \mathbb{R}$, $t : \mathbb{R}$	$\mathbb{R}$	OutOfDomain
TData_d2Tdy2	$y : \mathbb{R}$, $t : \mathbb{R}$	$\mathbb{R}$	OutOfDomain

9.4 Semantics

9.4.1 State Variables

S: sequence of PiecewiseT
Y: sequence of $\mathbb{R}$
t_L: sequence of $\mathbb{R}$

T_L: sequence of $\mathbb{R}$
t_S: sequence of $\mathbb{R}$
T_S: sequence of $\mathbb{R}$

9.4.2 State Invariants

None

9.4.3 Assumptions

TData_init() will be called before any other access programs.

9.4.4 Access Routine Semantics

TData_init():

- transition: $S, Y, t_L, T_L, t_S, T_S :=<>, <>, <>, <>$
- exception: none

TData_add($s, y, t_L^*, T_L^*, t_S^*, T_S^*$):

- transition: $S, Y, t_L, T_L, t_S, T_S := S||<s>, Y||<y>, t_L||<t_L^*>, T_L||<T_L^*>, t_S||<t_S^*>, T_S||<T_S^*>$
- exception: $exc := (|Y| > 0 \wedge y \leq Y_{|Y|-1} \Rightarrow \text{IndepNotAscending})$

TData_rm(i):

- transition: $s := s[0..i-1]||s[i+1..|s|-1]$
- exception: $exc := (\neg(0 \leq i < |S| - 1) \Rightarrow \text{InvalidIndex})$

TData_getC(i):

- output: $out := S[i]$
- exception: $exc := (\neg(0 \leq i < |S| - 1) \Rightarrow \text{InvalidIndex})$

TData_Liq(i):

- output: $out := t_L[i], T_L[i]$
- exception: $exc := (\neg(0 \leq i < |S| - 1) \Rightarrow \text{InvalidIndex})$

TData_Sol(i):

- output: $out := t_S[i], T_S[i]$
- exception: $exc := (\neg(0 \leq i < |S| - 1) \Rightarrow \text{InvalidIndex})$

TData_slice(t):

- output: $out := \langle Y_0, Y_1, ..., Y_{|Y|-1} \rangle, \langle S_0.\text{feval}(t), S_1.\text{feval}(t), ..., S_{|S|-1}.\text{feval}(t) \rangle$
- exception: None

TData_breakPts():

- output: $out := \langle Y_0, Y_1, ..., Y_{|Y|-1} \rangle, \langle S_0.x_1, S_1.x_1, ..., S_{|S|-1}.x_1 \rangle, \langle S_0.x_2, S_1.x_2, ..., S_{|S|-1}.x_2 \rangle$
- exception: None

TData_T(t, y):

- output: Find *out* using the following steps:
  1. Using TData_breakPts() and interpolation determine the approximate values for the breakpoints (t_1^{approx}, t_1^{approx}) for the curve at height y.
  2. Determine what segment of the piecewise curve applies at time t. It will be the first segment if $t \leq t_1^{\text{approx}}$, the second segment if $t_1^{\text{approx}} < t \leq t_2^{\text{approx}}$ and the third segment if $t_2^{\text{approx}} < t$.
  3. Use the curves in S that are in the same segment (first, second or third) at given by the t and y combination. Use the data from these curves to interpolate the required *out* value.
- exception: $exc := (\neg\text{isInBounds}(Y, y) \Rightarrow \text{OutOfDomain})$

dTdt(t, y):

- output: $out := \frac{T(y,t+\Delta t)-T(y,t)}{\Delta t}$
- exception: $exc := (\neg\text{isInBounds}(Y, y) \Rightarrow \text{OutOfDomain})$

d2Tdy2(t, y):

- output: $out := \frac{T(y-\Delta y,t)+T(y+\Delta y,t)-2*T(y,t)}{\Delta y^2}$
- exception: $exc := (\neg\text{isInBounds}(Y, y) \Rightarrow \text{OutOfDomain})$

10 MIS of Identify Points Module

10.1 Module

IdentifyPts

10.2 Uses

None

10.3 Syntax

10.3.1 Exported Access Programs

Name	In	Out	Exceptions
find_breakPts	x: seq of $\mathbb{R}$, y: seq of $\mathbb{R}$	$\mathbb{R}$, $\mathbb{R}$	
find_liquidus	s : string	seq of $\mathbb{R}$, seq of $\mathbb{R}$	
find_solidus	s : string	seq of $\mathbb{R}$, seq of $\mathbb{R}$	

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Assumptions

10.4.3 Access Routine Semantics

find_breakPts(T)

- output: Given the sequence T identify the two points where there is a discontinuity in the slope. Return these two points, which can then be used as the initial guesses for x_1 and x_2. Hopefully the points can be determined using the first derivative of the data and the peak detection utilities. [Investigation is necessary here. —SS]
- exception: None

find_liquidus(s):
From the input file,

- for i in range 0 to 7 , calculate the following

$$\text{tempdiff} = TC_{i+1} - TC_i$$

- for TC0, invert the tempdiff by $(0 - \text{tempdiff})$ and find the first peak using peakutils, the corresponding temp value at maximum $(0 - \text{tempdiff})$ is the solidus point for TC0.
- from TC1 the solidus point is defined as the last point at which the temp diff is maximum. please note this does not apply to first TC.
- Note down the temperature and time values.
- Modify the state of the variable T_S and t_S.

LiquidusPoint():

From the input file,

- calculate tempdiff as follows

$$\text{tempdiff} = TC_1 - TC_0$$

- find the peaks using the peakutils
- The temperature at the first peak in tempdiff is taken as the liquidus temperature for all the thermocouples.
- Note down the temperature and time values.
- Modify the state of the variable T_L and t_L.

liqTemp(y):

- Obtain a polynomial regression between location values and the T_L.
- Pass y to this polynomial to obtain liquidus temperature ($T_L(y)$).

liqtime(y):

- Obtain a polynomial regression between location values and the t_L.
- Pass y to this polynomial to obtain liquidus time ($t_L(y)$).

solTemp(y):

- Obtain a polynomial regression between location values and T_S.

- Pass y to this polynomial to obtain solidus temperature ($T_S(y)$).

soltime(y):

- Obtain a polynomial regression between location values and T_S.

- Pass y to this polynomial to obtain solidus time ($t_S(y)$).

11 MIS of Calculation Module

11.1 Module

calc [This module spec needs work. —SS]

11.2 Uses

expt, temp

11.3 Syntax

11.3.1 Exported Access Programs

Name	In	Out	Exceptions
calculate_h	temp	h(t)($\mathbb{R}^n$)	
calculate_α_b	temp	α_b ($\mathbb{R}$)	
calculate_α_e	temp	α_e ($\mathbb{R}$)	
calculate_FS	temp, L($\mathbb{R}$), $C_v^L(T)(\mathbb{R})$, $C_v^S(T)(\mathbb{R})$, $\rho_L(T)(\mathbb{R})$, $\rho_S(T)(\mathbb{R})$	$f_s(\mathbb{R}^n)$	bad_C_v^L, bad_C_v^S, bad_f_s

11.4 Semantics

11.4.1 State Variables

α_b: $\mathbb{R}$ [Should this module really have state variables? —SS]
α_e: $\mathbb{R}$
f_s: $\mathbb{R}^n$

11.4.2 Assumptions

calculate_h, calculate_α_b and calculate_α_e will be called before calling calculate_FS.

11.4.3 Access Routine Semantics

calc.$alpha_b$:

- output: $out := alpha_b$
- exception: none

calc.$alpha_e$:

- output: $out := alpha_e$

- exception: none

calc.f_s:

- output: $out := f_s$
- exception: none

calculate_h():

- output: $out :=$ h(t)
- exception: NA
- equation:
  find h(t) such that it satisfies the PDE and boundary condition below.

  PDE:

$$\frac{\partial T}{\partial t} = \alpha_S(T)\frac{\partial^2 T}{\partial y^2}, \text{ where } \alpha_S(T) = \frac{k_S(T)}{\rho_S(T)C_p^S(T)}$$

  Boundary condition:

  $q = 0$ on all boundaries, except for the bottom of the cylinder

$$q(t) = h(t)(T_0(t) - T_{\text{env}}(t))$$

- Reference:*from IM1 in SRS*

calculate_α_b():

- output: $out := \alpha_b$
- exception: none
- equation: find α_b such that the following PDE and boundary conditions are satisfied PDE:

$$\frac{\partial T}{\partial t} = \alpha_b\frac{\partial^2 T}{\partial y^2}$$

  Boundary condition:

  $q = 0$ on all boundaries, except for the bottom of the cylinder where,

$$q(t) = h(t)(T_0(t) - T_{\text{env}}(t))$$

- Reference *from IM2 in SRS*

calculate_α_e():

- output: $out := \alpha_e$
- exception: none
- equation: find α_e such that the following PDE and boundary conditions are satisfied

  PDE

$$\frac{\partial T}{\partial t} = \alpha_b \frac{\partial^2 T}{\partial y^2}$$

  Boundary condition:

  $q = 0$ on all boundaries, except for the bottom of the cylinder where,

$$q(t) = h(t)(T_0(t) - T_{\text{env}}(t))$$

- Reference: *from IM3 in SRS*

calculate_FS():

- output: $out := f_s$
- exception: bad_f_s
- equation: Solve $f_s(t)$ at location y^* such that the following ODE is satisfied with $f_s(t_L) = 0$:

  ODE:

$$\dot{f}_s(f_s, t) = \frac{C_v(f_s)}{L\rho(f_s)} \left[\frac{\partial T(t)}{\partial t} - \alpha(f_s) \frac{\partial^2 T(t)}{\partial y^2} \right]$$

  where
  $C_v(f_s) = C_v^b(1 - f_s) + C_v^e f_s$
  $\rho(f_s) = \rho_b(1 - f_s) + \rho_e f_s$
  $\alpha(f_s) = \alpha_b(1 - f_s) + \alpha_e f_s$

where
$C_v^b = C_v^L(T_L)$, $C_v^e = C_v^S(T_S)$, $\rho_b = \rho_L(T_L)$, $\rho_e = \rho_S(T_S)$.
After solving the ODE, $f(t)$ and $T(t)$ are known, which means that $f(T)$ can be shown.

- Reference: *from IM4 in SRS*

12 Load Module

12.1 Module

Load

12.2 Uses

PiecewiseADT, TData, config, IdentifyPts

12.3 Syntax

12.3.1 Exported Constants

None

Exported Access Programs

Routine name	In	Out	Exceptions
parse	s : string		
loadTData	x : $\mathbb{R}^n$, y: $\mathbb{R}^n$		ValueError

12.4 Semantics

12.4.1 Environment Variables

infile: two dimensional sequence of text characters

12.4.2 State Variables

None

12.4.3 State Invariant

None

12.4.4 Assumptions

The input file will match the expected specification.

12.4.5 Access Routine Semantics

parse(s)
loadTData(x,y)

- transition: read data from the file infile associated with the string s. Use this data to update the state of the TData module. Load will first initialize TData (TData.init()) before populating TData with piecewise curves, y value and liquidus and solidus values. In loading the data, the following should be kept in mind:
  - The initial transient data for each thermocouple will be ignored. The first data point for the thermocouples corresponds to the maximum temperature reading for the first (lowest value of y) thermocouple.
  - The time data should be the actual time, not the number of accumulated data points. This will mean multiplying by the time step size from the config file.
  - The data should be verified against the constraints in Table 2 (Input Variables for Calculating $T(y, t)$ in the SRS. That is, there should be more than zero data points and all temperature values should be within the prescribed bounds.
  - The IdentifyPts functions should be used to find the initial estimates for the breakpoints and the values of the liquidus and solidus times and temperatures.
- exception: none

13 MIS of Output Verification Module

13.1 Module

verifyop

13.2 Uses

13.3 Syntax

13.3.1 Exported Access Programs

Name	In	Out	Exceptions
verify_output	f_s: $\mathbb{R}_n$		bad_f_s

13.4 Semantics

13.4.1 State Variables

None

13.4.2 Assumptions

All of the fields of the input parameters structure have been assigned a value.

13.4.3 Access Routine Semantics

verify_output(f_s):

- exception: $\forall t | 0 \leq t \leq t_{\text{final}},\ f_s < 0$ or $f_s > 1 \rightarrow bad\_f_s$

14 MIS of Output Module

14.1 Module

op

14.2 Uses

-

14.3 Syntax

14.3.1 Exported Access Program

Name	In	Out	Exceptions
output_calib	fname: string, t:$\mathbb{R}^n$,$f_s(t)$:$\mathbb{R}^n$		
output_calc	fname: string, t:$\mathbb{R}^n$,$f_s(t)$:$\mathbb{R}^n$		

14.4 Semantics

14.4.1 State Variables

$f_s : \mathbb{R}_n$ (between 0 & 1)

14.4.2 Environment Variables

file1: A file - calibration
file2: A file - calculation

14.4.3 Access Routine Semantics

output_calib(fname, t,f_s):

- transition: Write to environment variable named fname the following: the time value and the corresponding f_s value..
- exception: none

output_calc(fname, t,f_s):

- transition: Write to environment variable named fname the following: the time value and the corresponding f_s value..
- exception: none

15 MIS of Piecewise Data Structure Module

15.1 Template Module

PiecewiseADT

15.2 Uses

Regression, SeqService

15.3 Syntax

Exported Types

PiecewiseT = ?

15.3.1 Exported Access Programs

Name	In	Out	Exceptions
PiecewiseT	$x : \mathbb{R}^n, y : \mathbb{R}^n$, $x_1^{\text{init}} : \mathbb{R}$, $x_2^{\text{init}} : \mathbb{R}$	PiecewiseT	IndepNotAscending, SeqSizeMismatch
minD		$\mathbb{R}$	
maxD		$\mathbb{R}$	
x_1	–	$\mathbb{R}$	
x_2	–	$\mathbb{R}$	
feval	$x : \mathbb{R}$	$\mathbb{R}$	OutOfDomain

15.4 Semantics

15.4.1 State Variables

x_1: $\mathbb{R}$ # *first breakpoint*
x_2: $\mathbb{R}$ # *second breakpoint*
f: $\mathbb{R} \to \mathbb{R}$ # *piecewise polynomial*
minx: $\mathbb{R}$
maxx: $\mathbb{R}$

State Invariant

None

15.4.2 Assumptions

None

15.4.3 Access Routine Semantics

new PiecewiseT($x : \mathbb{R}^n$, $y : \mathbb{R}^n$, $x_1^{\text{init}} : \mathbb{R}$, $x_2^{\text{init}} : \mathbb{R}$):

- transition: *# changing the values of x_1, x_2 and f according to the following steps using local the following local real variables a_1, b_1, c_1, d_1, a_2, b_2, c_2, a_3, b_3, c_3, d_3, e_3, f_3* [Should re-write this so that is less algorithmic (operational spec) and more generic (descriptive spec) —SS]

  1. Additional local variables for independent variable: $x_0 = x[0]$, $n = \text{size}(x)$, $x_4 = x[n-1]$, $x_3 = x[\text{indexOf}((x_4 - x_2)/2)]$ *# x_3 is midway between x_2 and x_4*
  2. minx $= x_0$
  3. maxx $= x_4$
  4. Additional local variables for dependent variable: $y_0 = y[0]$, $y_1 = y[\text{indexOf}(x_1, x)]$, $y_2 = y[\text{indexOf}(x_2, x)]$, $y_3 = y[\text{indexOf}(x_3, x)]$, $y_4 = y[n-1]$
  5. Initial guess for parameters for first polynomial (assume linear): $a_1 = b_1 = 0; d_1 = y_0; c_1 = \frac{y_1 - y_0}{x_1}$
  6. Initial guess for parameters for second polynomial (assume linear): $a_2 = b_2 = 0; d_2 = y_1; c_2 = \frac{y_2 - y_1}{x_2 - x_1}$
  7. Initial guess for parameters for third polynomial (assume quadratic): $a_3 = b_3 = c_3 = d_3 = e_3 = 0$; $e_3 = \frac{(y_3 - y_2)(x_4 - x_2) - (y_4 - y_2)(x_3 - x_2)}{(x_3 - x_2)^2(x_4 - x_2) - (x_4 - x_2)^2(x_3 - x_2)}$; $f_3 = \frac{(y_3 - y_2)^2(y_4 - y_2) - (x_4 - x_2)^2(y_3 - y_2)}{(x_3 - x_2)^2(x_4 - x_2) - (x_4 - x_2)^2(x_3 - x_2)}$; $g_3 = y_2$
  8. $p^{\text{init}} = [x_1, x_2, a_1, b_1, c_1, d_1, a_2, b_2, c_2, a_3, b_3, c_3, d_3, e_3, f_3]$
  9. $p =$ optimize(ftrial, x, y, p^{init})
  10. $x_1 = p[0]$
  11. $x_2 = p[1]$
  12. $f = \lambda x :$ ftrial(x, p)

- output: *out* := self
- exception:
- exception: ($\neg$isAscending(x) $\Rightarrow$ IndepNotAscending$||x| \neq |y| \Rightarrow$ SeqSizeMismatch)

[This specification should be changed to allow the order of the polynomial to vary in each segment. —SS]

minD():

- output: *out* := minx
- exception: None

maxD():

- output: *out* := maxx
- exception: None

x_1:

- output: *out* := x_1
- exception: none

x_2:

- output: *out* := x_2
- exception: none

feval:

- output: *out* := $f(x)$
- exception: $(\neg(\text{minx} \leq x \leq \text{maxx}) \Rightarrow \text{OutOfDomain}))$

15.5 Local Functions

ftrial(x, x_1, x_2, a_1, b_1, c_1, d_1, a_2, b_2, c_2, a_3, b_3, c_3, d_3, e_3, f_3)
$p_1 = \lambda x : a_1x^3 + b_1x^2 + c_1x + d_1$
$d_2 = p_1(x_1)$
$p_2 = \lambda x : a_2x^3 + b_2x^2 + c_2x + d_2$
$g_3 = p_2(x_2)$
$p_2 = \lambda x : a_3x^6 + b_3x^5 + c_3x^4 + d_3x^3 + e_3x^2 + f_3x + g_3$
return $((x < x_1) \Rightarrow p_1(x) | (x_1 \leq x < x_2) \Rightarrow p_2(x) | (x \geq x_2) \Rightarrow p_3(x))$

indexOf($x^* : \mathbb{R}$, $x : \mathbb{R}^n$)
indexOf(x^*, x) $\equiv i$ such that $x[i] \leq x^* \leq x[i+1]$ # *maybe select closest x instead?*

Considerations

The fitting for the piecewise curve depends on determining a good initial guess for the parameters. Section 20.2 provides an overview of how the initial guess can be determined.

Sequence Services Module

Module

SeqServices

Uses

None

Syntax

Exported Constants

None

Exported Access Programs

Routine name	In	Out	Exceptions
isAscending	$X : \mathbb{R}^n$	$\mathbb{B}$	
isInBounds	$X : \mathbb{R}^n, x : \mathbb{R}$	$\mathbb{B}$	

Semantics

State Variables

None

State Invariant

None

Assumptions

None, unless noted with a particular access program

Access Routine Semantics

isAscending(X)

- output: $out := \neg\exists(i|i \in [0..|X|-2] : X_{i+1} < X_i)$
- exception: none

isInBounds(X, x) # *assuming isAscending is True*

- output: $out := X_0 \leq x \leq X_{|X|-1}$
- exception: none

16 MIS of Interpolation Module

This module follows the interface for intepolation routines provided by numpy and scipy in Python.

17 MIS of Linear Regression Module

This module follows the interface for regression routines provided by numpy and scipy in Python.

18 MIS of ODE Solver Module

#Bold font is used to indicate variables that are a sequence type

18.1 Module

Solver($n : \mathbb{N}$) *#n is the length of the sequences*

18.2 Uses

None

18.3 Syntax

18.3.1 Exported Access Programs

Name	In	Out	Except.
solve	$\mathbf{f} : (\mathbb{R}^{n+1} \to \mathbb{R})^n, t_0 : \mathbb{R}, \mathbf{y}_0 : \mathbb{R}^n, g : \mathbb{R}^n \to \mathbb{R}, t_{\text{fin}} : \mathbb{R}$	$t_1 : \mathbb{R}, \mathbf{y} : (\mathbb{R} \to \mathbb{R})^n$	ODE_ERR, NO_EVENT
solveNoE	$\mathbf{f} : (\mathbb{R}^{n+1} \to \mathbb{R})^n, t_0 : \mathbb{R}, \mathbf{y}_0 : \mathbb{R}^n, t_{\text{fin}} : \mathbb{R}$	$\mathbf{y} : (\mathbb{R} \to \mathbb{R})^n$	ODE_ERR

18.4 Semantics

18.4.1 State Variables

None

18.4.2 Access Routine Semantics

#Solving $\frac{d}{dt}\mathbf{y} = \mathbf{f}(t, \mathbf{y}(t))$

solve($\mathbf{f}, t_0, \mathbf{y}_0, g, t_{\text{fin}}$):

- output: $out := t_1, \mathbf{y}(t)$ where

$$\mathbf{y}(t) = \mathbf{y}_0 + \int_{t_0}^{t} \mathbf{f}(s, \mathbf{y}(s))ds$$

  with t_1 determined by the first time where $g(\mathbf{y}(t_1)) = 0$. $\mathbf{y}(t)$ is calculated from $t = t_0$ to $t = t_1$.

- exception: $exc :=$ ($\neg(\exists t : \mathbb{R} | t_0 \leq t \leq t_{\text{fin}} : g(\mathbf{y}(t)) = 0) \Rightarrow$ NO_EVENT | ODE Solver Fails $\Rightarrow$ ODE_ERR)

solveNoE($\mathbf{f}$, t_0, $\mathbf{y}_0$, t_{fin}):

- output: $out := \mathbf{y}(t)$ where

$$\mathbf{y}(t) = \mathbf{y}_0 + \int_{t_0}^{t_{\text{fin}}} \mathbf{f}(s, \mathbf{y}(s))ds$$

  $y(t)$ is calculated from $t = t_0$ to $t = t_{\text{fin}}$.

- exception: $exc :=$ (ODE Solver Fails $\Rightarrow$ ODE_ERR)

19 MIS of Plot Module

19.1 Module

plot

19.2 Uses

N/A

19.3 Syntax

19.3.1 Exported Access Programs

Name	In	Out	Exceptions
PlotSeq	$X : \mathbb{R}^n, Y : \mathbb{R}^n$		SeqSizeMismatch

19.4 Semantics

19.4.1 State Variables

None

19.4.2 Environment Variables

win: 2D sequence of pixels displayed on the screen

19.4.3 Assumptions

None

19.4.4 Access Routine Semantics

PlotSeq(X, Y)

- transition: modify win so that it displays an x-y graph of the data points showing X and the corresponding Y values. X is the independent variable and Y as the dependent variable.
- exception: ($|X| \neq |Y| \Rightarrow$ SeqSizeMismatch)

20 Appendix

[we don't have a range for dy, α_S, α_L, h,Do we need them? —Malavika]

20.1 Exceptions

Table 3: Possible Exceptions

Message ID	Error Message
bad_H:	H must be > 0.001 and < 100
bad_D:	D must be > 0.001 and < 100
bad_n:	n should be $< 1 \times 10^4$
bad_dt:	dt must be > 0.0001 and < 1000
bad_dy:	
bad_k_S:	k_S should be > 0.001 and $< 1 \times 10^5$
bad_k_L:	k_L should be >0.001 and $< 1 \times 10^5$
bad_C_p^L:	C_p^L should be $> 1 \times 10^{-4}$ and $< 1 \times 10^4$
bad_C_p^S:	C_p^S should be $> 1 \times 10^{-4}$ and $< 1 \times 10^4$
bad_C_v^L:	C_v^L should be $> 1 \times 10^{-4}$ and $< 1 \times 10^4$
bad_C_v^S:	C_v^S should be $> 1 \times 10^{-4}$ and $< 1 \times 10^4$
bad_ρ_S:	ρ_S should be > 0.001 and $< 1 \times 10^4$
bad_ρ_L:	ρ_L should be > 0.001 and $< 1 \times 10^4$
bad_α_S:	α_S should be > 0.001 and $< 1 \times 10^4$
bad_α_L:	α_L should be > 0.001 and $< 1 \times 10^4$
bad_T_{env}:	T_{env} should be between 0 and 100
bad_h:	[We dont have a range for this —Malavika]
bad_f_s:	f_s must be between 0 and 1

[α_S and α_L were not in parameter specification table, so I gave it values similar to ρ. If this is not necessary, it can be removed. From SRS, α is computed. So, I think it is not necessary —Malavika]

20.2 Derivation of Piecewise Curve Fitting

The equation for the three piece regression is as follows:

$p_1(t) = a_1t^3 + b_1t^2 + c_1t + d_1$
$p_2(t) = a_2(t - t_1)^3 + b_2(t - t_1)^2 + c_1(t - t_1) + d_2$

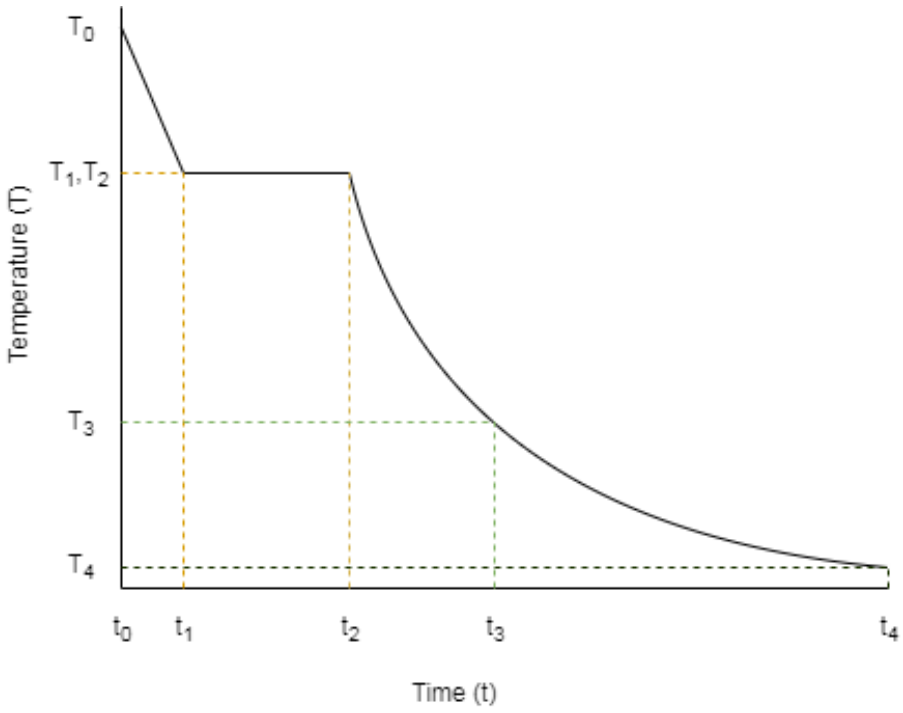

Figure 1: Piecewise regression breakdown of a thermocouple

$$p_3(t) = a_3(t-t_2)^6 + b_3(t-t_2)^5 + c_3(t-t_2)^4 + d_3(t-t_2)^3 + e_3(t-t_2)^2 + f_3(t-t_2) + g_3$$

where
$d_1 = T_0$
$p_1(t_1) = p_2(t_1)$ so $d_2 = p_1(t_1) = a_1{t_1}^3 + b_1{t_1}^2 + c_1t_1 + d_1$
$p_2(t_2) = p_3(t_2)$ so $f_3?/g_3? = p_2(t_2) = a_2(t_2-t_1)^3 + b_2(t_2-t_1)^2 + c_1(t_2-t_1) + d_2$

Initial Piecewise Regression Equation Guesses

Guess between t_0 to t_1

$p_1(t) = c_1t + d_1$ where $p_1(t_0) = T_0$
substituting $d_1 = T_0$, $p_1(t) = c_1t + T_0$

$p_1(t_1) = T_1$ so $c_1t_1 + T_0 = T_1$

$$c_1 = \frac{T_1 - T_0}{t_1}$$

$$\therefore p_1(t) = \frac{T_1 - T_0}{t_1}t + T_0$$

Guess between t_1 to t_2

$p_2(t) = c_2(t - t_1) + d_2$ where $d_2 = p_1(t_1)$
$= c_2(t - t_1) + p_1(t_1)$

$p_2(t_2) = T_2$ so $c_2(t_2 - t_1) + p_1(t_1) = T_2$

$$c_2 = \frac{T_2 - p_1(t_1)}{t_2 - t_1}$$

$$\therefore p_2(t) = \frac{T_2 - p_1(t_1)}{t_2 - t_1}(t - t_1) + p_1(t_1)$$

Guess between t_2 to t_4

$p_3(t) = e_3(t - t_2)^2 + f_3(t - t_2) + g_3$

$p_3(t_2) = T_2 = p_2(t_2) = g_3$
$p_3(t_3) = T_3 = e_3(t_3 - t_2)^2 + f_3(t_3 - t_2) + g_3$
$p_3(t_4) = T_4 = e_3(t_4 - t_2)^2 + f_3(t_4 - t_2) + g_3$

Solving for e_3, f_3 with the given matrix:

$$\begin{bmatrix} (t_3 - t_2)^2 & (t_3 - t_2) \\ (t_4 - t_2)^2 & (t_4 - t_2) \end{bmatrix} \begin{bmatrix} e_3 \\ f_3 \end{bmatrix} = \begin{bmatrix} (T_3 - f_3) \\ (T_4 - f_3) \end{bmatrix} = \begin{bmatrix} (T_3 - T_2) \\ (T_4 - T_2) \end{bmatrix}$$

Using Cramer's Rule $Ax = b$:

$$A = \begin{vmatrix} (t_3 - t_2)^2 & (t_3 - t_2) \\ (t_4 - t_2)^2 & (t_4 - t_2) \end{vmatrix} = (t_3 - t_2)^2(t_4 - t_2) - (t_4 - t_2)^2(t_3 - t_2)$$

$$B = \begin{vmatrix} T_3 - T_2 & (t_3 - t_2) \\ T_4 - T_2 & (t_3 - t_2) \end{vmatrix} = (T_3 - T_2)(t_4 - t_2) - (T_4 - T_2)(t_3 - t_2)$$

$$C = \begin{vmatrix} (t_3 - t_2)^2 & (T_3 - T_2) \\ (t_4 - t_2)^2 & (T_4 - T_2) \end{vmatrix} = (t_3 - t_2)^2(T_4 - T_2) - (t_4 - t_2)^2(T_3 - T_2)$$

Finding e_3

$$e_3 = \frac{B}{A} = \frac{(T_3 - T_2)(t_4 - t_2) - (T_4 - T_2)(t_3 - t_2)}{(t_3 - t_2)^2(t_4 - t_2) - (t_4 - t_2)^2(t_3 - t_2)}$$

Finding f_3

$$f_3 = \frac{C}{A} = \frac{(t_3 - t_2)^2(T_4 - T_2) - (t_4 - t_2)^2(T_3 - T_2)}{(t_3 - t_2)^2(t_4 - t_2) - (t_4 - t_2)^2(t_3 - t_2)}$$

Table 2: Specification Parameter Values

Var	**Value**
AbsZero	273.15 °C
$H_{\min}$	0.001 m
$H_{\max}$	100 m
$D_{\min}$	0.001 m
$D_{\max}$	100 m
$m_{\max}$	1000
$dt_{\min}$	0.0001 s
$dt_{\max}$	1000 s
$n_{\max}$	1×10^4
$T_{\min}$	100 °C
$T_{\max}$	1×10^4 °C
$k_{S\min}$	0.001 W/(m °C)
$k_{S\max}$	1×10^5 W/(m °C)
$k_{L\min}$	0.001 W/(m °C)
$k_{L\max}$	1×10^5 W/(m °C)
$\rho_{S\min}$	0.001 kg m^{-3}
$\rho_{S\max}$	1×10^4 kg m^{-3}
$\rho_{L\min}$	0.001 kg m^{-3}
$\rho_{L\max}$	1×10^4 kg m^{-3}
$C_{P\min}^{L}$	1×10^{-4} J kg^{-1} °C^{-1}
$C_{P\max}^{L}$	1×10^4 J kg^{-1} °C^{-1}
$C_{P\min}^{S}$	1×10^{-4} J kg^{-1} °C^{-1}
$C_{P\max}^{S}$	1×10^4 J kg^{-1} °C^{-1}
$C_{v\min}^{L}$	1×10^{-4} J kg^{-1} °C^{-1}
$C_{v\max}^{L}$	1×10^4 J kg^{-1} °C^{-1}
$C_{v\min}^{S}$	1×10^{-4} J kg^{-1} °C^{-1}
$C_{v\max}^{S}$	1×10^4 J kg^{-1} °C^{-1}
$L_{\min}$	0.001 m
$L_{\max}$	1×10^{10} m
$T_{\text{env}\min}$	0 °C
$T_{\text{env}\max}$	100 °C