Module Interface Specification for SCEC (Solar Cooker Energy Calculator)

Deesha Patel

March 19, 2023

1 Revision History

Date	Version	Notes
March 17, 2023	0.1	Initial Release
March 19, 2023	0.2	Updates according to the comments

2 Symbols, Abbreviations and Acronyms

See SRS Documentation for symbols, abbreviations and acronyms.

symbol	description
c	Condition
en	Energy
energySeq	Energy Sequence
ODEs	Ordinary Differential Equations
param	Parameters
r	Rule
SCEC	Solar Cooker Energy Calculator
temp	Temperature value
tempSeq	Temperature Sequence

Contents

3 Introduction

The following document details the Module Interface Specifications for SCEC (Solar Cooker Energy Calculator). This document specifies how every module is interfacing with every other parts.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at Github repository for SCEC.

4 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 SCEC.

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 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, uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 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	Constant Value Module Energy Equation Module Input Format Module Input Parameter Module Output Format Module Control Module Temperature ODE Module
Software Decision Module	ODE Solver Module Plotting Result Module Sequence Data Structure Module

Table 1: Module Hierarchy

6 MIS of Constant Value Module

6.1 Module

ConstValueParams

6.2 Uses

- Hardware Hiding Module

6.3 Syntax

6.3.1 Exported Constants

params.stefan_const	:=	5.670374419e-08	#Stefan-Boltzman constant
params.h_t_int3	:=	4.0	#Heat flux from Lid to Inner area of container
params.h_ref_int2	:=	4.4	#Heat flux from reflector to Inner area of box
params.h_ref_f	:=	4.0	#Heat flux from reflector to fluid
params.c_ref	:=	900	#Specific heat capacity of reflector
params.c_f	:=	4190	#Specific heat capacity of fluid
params.m_ref	:=	0.2	#Mass of reflector
params.m_f	:=	2.0	#Mass of fluid
params.t_g	:=	0.48	#Transmittivity of glass
params.p	:=	0.89	#Reflectivity of glass

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
ConstValueParams	-	-	-

6.4 Semantics

6.4.1 State Variables

params: An object of ConstValueParams contains real values.

- params.stefan_const : $\mathbb{R}$
- params.h_t_int3 : $\mathbb{R}$
- params.h_ref_int2 : $\mathbb{R}$

- params.h_ref_f : $\mathbb{R}$
- params.c_ref : $\mathbb{R}$
- params.c_f : $\mathbb{R}$
- params.m_ref : $\mathbb{R}$
- params.m_f : $\mathbb{R}$
- params.t_g : $\mathbb{R}$
- params.p : $\mathbb{R}$

6.4.2 Environment Variables

None

6.4.3 Assumptions

None

6.4.4 Access Routine Semantics

None

6.4.5 Local Functions

None

6.4.6 Considerations

Note: These constants are as per the SRS document, so constant parameters and values may change according to the implementation if required.

7 MIS of Energy Equation Module

7.1 Module

energy_calculation

7.2 Uses

- Input Parameter Module
- Constant Value Module

7.3 Syntax

7.3.1 Exported Constants

None

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
energyWat	sequence of $\mathbb{R}$	sequence of $\mathbb{R}$	MissingParamError, TempValueError, EnergyValueError, EnergySeqError, TempSeqError

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Environment Variables

None

7.4.3 Assumptions

The Energy Equation Module is called through the Control Module, ensuring that Temperature ODE Module has been called before Energy Equation Module and fluid temperature values are calculated to give input to the Energy Equation Module.

7.4.4 Access Routine Semantics

This module satisfies R5 from the SRS.

energyWat(temp):

- transition: None
- output: The energy equation module returns the sequence of energy:
  out := $\{\ i \in \mathbb{N}, 0 \leq i < |s-1|\ |\ m_f c_f(temp[i]-T_{\text{init}})\ \}$, where $m_f = getConstantValue(m\_f)$, $c_f = getConstantValue(c\_f)$
- exception: exc :=

Expression	Exception	Description
$\neg(\triangle T > 0)$	TempValueError	Valid temperature value should positive only.
$(E_f = 0 \vee E_f \notin \mathbb{R})$	EnergyValueError	Energy of fluid should be real and non zero number.
$(en = \emptyset)$	EnergySeqError	Energy sequence should have at least one value, not an empty sequence.
$(temp = \emptyset)$	TempSeqError	Temperature sequence should not be empty.

7.4.5 Local Functions

getConstValue : String
getConstValue(param): A function to fetch the mass and capacity of fluid from the Constant Value Module.

- output: out:=
  m_f : $\mathbb{R}$,
  c_f : $\mathbb{R}$
- exception: exc:=

Expression	Exception	Description
$(m_f = \nexists \vee c_f = \nexists)$	MissingParamError	Valid string should be available in the Constant Value Module.

8 MIS of Input Format Module

8.1 Module

format_input

8.2 Uses

- Input Parameter Module
- Hardware Hiding Module

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
load_params	String	sequence of $\mathbb{R}$	various (See table 2)

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

1. paramFile: A file containing sequence of strings that provides data related to temperature, Area and other properties.

8.4.3 Assumptions

- The Control Module call this module for formating input parameters.
- The paramFile contains input starts with '#' in new line. The order of the inputs should be as below:
  Line 1: Area of lid
  Line 2: Temperature of lid
  Line 3: Temperature of fluid
  Line 4: Emissivity of lid
  Line 5: Area of reflector

Line 6: Temperature of reflector
Line 7: Emissivity of reflector
Line 8: Temperature of glass
Line 9: Area of mass

8.4.4 Access Routine Semantics

This module is a function to load, verify and store input data. (R1 and R2 from SRS).

load_params(paramFile):

- transition: paramFile is the file for fetching input values from the file. The following procedure is performed:
  1. Verify the format of the file to be .txt.
  2. Extract the input one by one.
  3. Verify all inputs, verifyInput(param)
  4. Store inputs to the data structure
- output: Give sequence of inputs contains all inputted data under appropriate field names.
  out := params
- exception: Data input which does not comply with the data constraints specified in SRS for this project will yield one of the potential exceptions or warning as listed in the appendix of this document.

8.4.5 Local Functions

verifyInputs : $\mathbb{R}$
verifyInputs(param): A function to verify the inputs for SCEC.

- output: out:= param
- exception: exc:= See appendix (Table 2) for all constraints and error message.

9 MIS of Input Parameter Module

9.1 Module

parameters

9.2 Uses

- Hardware Hiding Module

9.3 Syntax

9.3.1 Exported Constants

None

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
__init__	-	-	-

9.4 Semantics

Parameters is a data structure designed to store the input information entered by the Input Format Module.

9.4.1 State Variables

param := sequence of (
$A_t : \mathbb{R}$, Area of lid
$T_t : \mathbb{R}$, Temperature of lid
$T_f : \mathbb{R}$, Temperature of fluid
$e_t : \mathbb{R}$, Emissivity of lid
$A_{\text{ref}} : \mathbb{R}$, Area of reflector
$T_{\text{ref}} : \mathbb{R}$, Temperature of reflector
$e_{\text{ref}} : \mathbb{R}$, Emissivity of reflector
$T_g : \mathbb{R}$, Temperature of glass
$A_m : \mathbb{R}$, Area of mass
)

9.4.2 Environment Variables

1. Windows screen: Input Format Module takes the input using showing it on screen.

2. Windows keyboard: Input Format Module takes the input from the keyboard in the file.

9.4.3 Assumptions

None

9.4.4 Access Routine Semantics

Parameters:

- transition: This module is a simple data structure for storing the input values formatted by Input Format Module.
- output: None
- exception: exc:= None

9.4.5 Local Functions

None

10 MIS of Output Format Module

10.1 Module

output

10.2 Uses

- Input Parameter Module
- Hardware Hiding Module
- Plotting Result Module

10.3 Syntax

10.3.1 Exported Constants

None

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
output	String, sequence of $\mathbb{R}$, sequence of $\mathbb{R}$, sequence of $\mathbb{N}$	Output File	MissingValueError, FileAlreadyExistError

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Environment Variables

1. file: The file in which the output is saved.
2. Window screen: Output Format Module prints the result in the graph, which is shown to the screen.

10.4.3 Assumptions

The Control Module properly verified values against the constraint.

10.4.4 Access Routine Semantics

output(fileName, tempSeq, energySeq, t):

- transition: None
- output: This module is able to output the file which contains output of the temperature and energy sequence.
  out := file
- exception: exc:= None

10.4.5 Local Functions

None

11 MIS of Control Module

11.1 Module

main

11.2 Uses

- Constant Value Module
- Energy Equation Module
- Hardware Hiding Module
- Input Format Module
- Output Format Module
- Temperature ODE Module
- ODE Solver Module
- Sequence Data Structure Module

11.3 Syntax

11.3.1 Exported Constants

None

11.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
main	-	Modifies output file	Various

11.4 Semantics

11.4.1 State Variables

- init_temp : sequence of [T1: $\mathbb{R}$, T2: $\mathbb{R}$] # initial temperature values.
- t : vector # vector of time.
- temp : sequence of [T1: $\mathbb{R}$, T2: $\mathbb{R}$] # sequence 2D for temperatures.
- e_f : sequence of [E1: $\mathbb{R}$] # sequence 1D for energy of fluid.

11.4.2 Environment Variables

None

11.4.3 Assumptions

None

11.4.4 Access Routine Semantics

main():

- transition: Control the order of execution of different modules as follow:

  - Set constant value using Constant Value Module (M2, Section 6).
  - Set inputted values to the appropriate variables using Input Format Module (M4, Section 8).
  - Set the time vector using Sequence Data Structure Module (M11, Section **??**).
  - Temperature values for reflector and fluid is calculated using initial conditions by Temperature ODE Module (M8, Section 12).
  - Using the previous step output, energy of fluid is calculated in Energy Equation Module (M3, Section 7).
  - Output is transferred to the output file with the help of Output Format Module and internally it also called Plotting Result Module for plotting result on graphs (M6, Section 10 and M10, Section 14).

- output: Main program request the Output Format Module at the end for producing file with plotted result.
- exception: exc:= Potential exceptions occurs are from different sub-modules only.

11.4.5 Local Functions

None

12 MIS of Temperature ODE Module

12.1 Module

calculation

12.2 Uses

- Constant Value Module
- Input Parameter Module

12.3 Syntax

12.3.1 Exported Constants

None

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
calculateOde	sequence of $[T1 : \mathbb{R}, T2 : \mathbb{R}]$, sequence of $\mathbb{R}$	sequence of $[T1 : \mathbb{R}, T2 : \mathbb{R}]$	TypeError, NameError, MissingValueError, EmptyArrayError, ValueError

12.4 Semantics

12.4.1 State Variables

- in_t: $\mathbb{R}$ # Calculate and store inner temperature of the box.
- $q11$: $\mathbb{R}$ # Heat flow convection of the lid toward the inner box.
- $q12$: $\mathbb{R}$ # Heat flow radiation of the lid of the recipient toward the fluid.
- $q13$: $\mathbb{R}$ # Heat flow convection of recipient to the inner box.
- $q14$: $\mathbb{R}$ # Heat flow reflection of incident radiation on the reflector.
- $q15$: $\mathbb{R}$ # Heat flow radiation of recipient toward glass2.
- $q16$: $\mathbb{R}$ # Heat flow radiation of recipient toward the fluid.

- $q17$: $\mathbb{R}$ # Heat flow convection of recipient toward the fluid.
- dr: $\mathbb{R}$ # Temperature of reflector.
- df: $\mathbb{R}$ # Temperature of fluid.

12.4.2 Environment Variables

None

12.4.3 Assumptions

None

12.4.4 Access Routine Semantics

calculation(initial_condition, params):

- transition: Temperature is calculated as follows:
  - Calculate and set the value of in_t which is an inner temperature of the box calculate by performing mean of glass, lid of recipient and reflector temperature.
  $$in\_t = \frac{T_{\text{glass}}+T_{\text{lid}}+T_{\text{ref}}}{3}$$
  - Calculate and store the values of qs using the input $params$.
  $q11 = A_t h_{\text{t-int3}}(T_{\text{t}} - T_f)$
  $q12 = A_t \sigma \epsilon_t (T_{\text{t}}^4 - T_f^4)$
  $q13 = A_{\text{ref}} h_{\text{ref-int2}}(T_{\text{int2}} - T_{\text{ref}})$
  $q14 = \sum_{i=1}^{n} \rho A_{\text{ref,n}} G \tau_g^2 cos(90 - \theta_{\text{ref,n}})$
  $q15 = A_{\text{ref}} \sigma \epsilon_{\text{ref}} (T_{\text{ref}}^4 - T_{\text{g2}}^4)$
  $q16 = A_{\text{ref}} \sigma \epsilon_{\text{ref}} (T_{\text{ref}}^4 - T_f^4)$
  $q17 = A_m h_{\text{ref-f}}(T_{\text{ref}} - T_f)$
  - Find the value of dr and df using qs and constant values.
  $$dr = \frac{q13+4q14-q15-q16-q17}{m_r c_r}$$
  $$df = \frac{q11+q12+q16+q17}{m_f c_f}$$
  - Return calculated dr and df as a sequence.
- output: Temperature ODE Module give an output of 2D sequence which stores temperature of reflector and fluid.
  out := s # temperature of reflector and fluid respectively

- exception: exc:=

Expression	**Exception**	**Description**
$(\forall i \in [0..\|s\|-1])(\text{initial\_condition}[i] \notin \mathbb{R})$	TypeError	Valid initial input for the temperature sequence are real numbers.
If any of the input is missing	MissingValueError	Module requires 3 input values: fileName, temperatureSeq, and energySeq
If tries to use variable that is not declared.	NameError	Variables those are declared in the module can accessible.

12.4.5 Local Functions

verifyTemp : $\mathbb{R}$
verifyTemp(temp): A function to verify the temperature sequence.

- output: None
- exception:

Expression	**Exception**	**Description**
$(dr < 0 \vee df < 0)$	ValueError	Valid temperature value should not negative
$(dr = \emptyset \vee df = \emptyset)$	EmptyArrayError	Temperature sequence should not null

13 MIS of ODE Solver Module

13.1 Module

solver

13.2 Uses

- Temperature ODE Module
- Sequence Data Structure Module

13.3 Syntax

13.3.1 Exported Constants

None

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
solveOde	String, sequence of $\mathbb{R}$, vector, sequence of $\mathbb{R}$	sequence of $[T1 : \mathbb{R}, T2 : \mathbb{R}]$	ValueError, TypeError, OverflowError, RuntimeError

13.4 Semantics

13.4.1 State Variables

None

13.4.2 Environment Variables

None

13.4.3 Assumptions

All input parameters to the $solveOde()$ are correct and verified by the Control Module.

13.4.4 Access Routine Semantics

solveOde($funcName, init\_cond, t, args$):

- transition: ODE is calculated as follows:

  - Takes specified inputs as a parameter.
  - With specified function name (first argument) in solveOde, initial conditions, time interval and extra parameters the solution is to be computed.
  - Output is store in the local variable.

- output: ODE Solver Module give an output of 2D sequence from Temperature ODE Module using programming library.
  - out := sequence s

- exception: exc:=

Expression	**Exception**	**Description**
$(\forall i \in [0..\|s\|-1])(\text{initial\_condition}[i] \notin \mathbb{R} \vee \text{initial\_condition}[i] \notin \mathbb{N} \vee \text{initial\_condition}[i] \notin \mathbb{Z})$	ValueError	Valid initial input for the temperature sequence are real or natural numbers.
solveOde(init_cond, funcName, t, args)	TypeError	Module requires 4 input values in order of funcName, init_cond, t and args.
If solution of solveOde results larger value of temperature than range of double.	OverflowError	Limit of the temperature should be correct.
If the specified function has some problems	RuntimeError	Function should work properly in order to solve the integration of ODE.

13.4.5 Local Functions

None

14 MIS of Plotting Result Module

14.1 Module

plot

14.2 Uses

- Hardware Hiding Module

14.3 Syntax

14.3.1 Exported Constants

None

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
plot	vector, sequence of $\mathbb{R}$	graph	TypeError

14.4 Semantics

14.4.1 State Variables

None

14.4.2 Environment Variables

Windows screen: As this module display a graph on screen, it uses screen for it.

14.4.3 Assumptions

None

14.4.4 Access Routine Semantics

plot(t, s):

- transition: None
- output: Plotting Result Module display the graph using the received input parameters. output := graph
- exception: exc:= None

14.4.5 Local Functions

None

14.5 Considerations

This module usually handle by the programming language. For SCEC system, we are using matplotlib to plot the result. So, exceptions are handled by the language itself.

15 MIS of Sequence Data Structure Module

15.1 Module

sequential

15.2 Uses

None

15.3 Syntax

15.3.1 Exported Constants

None

15.3.2 Exported Access Programs

None

15.4 Semantics

15.4.1 State Variables

None

15.4.2 Environment Variables

None

15.4.3 Assumptions

None

15.4.4 Access Routine Semantics

None

15.4.5 Local Functions

None

15.5 Considerations

The Sequence Data Structure Module is handled by the programming language. For the purpose of sequences, SCEC is using NumPy.

16 Appendix

Table 2: Possible errors for input

Var	Physical Constraints	Error Message
A_{ref}	$0 < A_{\text{ref}} \leq 1$	InvalidInputError
A_m	$0 < A_m \leq 1$	InvalidInputError
A_t	$0 < A_t \leq 1$	InvalidInputError
T_{f}	$20 < T_{\text{f}} < 100$	InvalidInputError
T_{ref}	$20 < T_{\text{ref}} < 100$	InvalidInputError
T_{g2}	$20 < T_{\text{g2}} < 100$	InvalidInputError
T_{t}	$20 < T_{\text{t}} < 100$	InvalidInputError
ϵ_{ref}	$0 < \epsilon_{\text{ref}} < 1$	InvalidInputError
ϵ_{t}	$0 < \epsilon_{\text{t}} < 1$	InvalidInputError

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.